Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 154 (2006) 640–649
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Hybrid rounding techniques for knapsack problems

Monaldo Mastrolilli, Marcus Hutter

IDSIA, Galleria 2, 6928 Manno, Switzerland

Received 21 December 2001; received in revised form 23 August 2005; accepted 23 August 2005
Available online 8 November 2005

Abstract

We address the classical knapsack problem and a variant in which an upper bound is imposed on the number of items that can be selected. We show that appropriate combinations of rounding techniques yield novel and more powerful ways of rounding. Moreover, we present a linear-storage polynomial time approximation scheme (PTAS) and a fully polynomial time approximation scheme (FPTAS) that compute an approximate solution, of any fixed accuracy, in linear time. These linear complexity bounds give a substantial improvement of the best previously known polynomial bounds [A. Caprara, et al., Approximation algorithms for knapsack problems with cardinality constraints, European J. Oper. Res. 123 (2000) 333–345].

Keywords: Knapsack problems; Approximation schemes; Arithmetic and geometric rounding; Dynamic programming

1. Introduction

In the classical *knapsack problem* (KP) we have a set $N := \{1, \ldots, n\}$ of items and a knapsack of limited capacity. To each item we associate a positive profit p_j and a positive weight w_j. The problem calls for selecting the set of items with maximum overall profit among those whose total weight does not exceed the knapsack capacity $c > 0$. KP has the following integer linear programming (ILP) formulation:

$$\text{maximize} \quad \sum_{j \in N} p_j x_j, \tag{1}$$

$$\text{subject to} \quad \sum_{j \in N} w_j x_j \leqslant c, \tag{2}$$

$$x_j \in \{0, 1\}, \quad j \in N, \tag{3}$$

where each binary variable x_j, $j \in N$, is equal to 1 if and only if item j is selected. In general, we cannot take all items because the total weight of the chosen items cannot exceed the knapsack capacity c. In the sequel, without loss of generality, we assume that $\sum_{j \in N} w_j > c$ and $w_j \leqslant c$ for every $j \in N$.

E-mail addresses: monaldo@idsia.ch (M. Mastrolilli), marcus@idsia.ch (M. Hutter).

0166-218X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2005.08.004

The *k-item knapsack problem* (kKP), is a KP in which an upper bound of k is imposed on the number of items that can be selected in a solution. The problem can be formulated as (1)–(3) with the additional constraint

$$\sum_{j\in N} x_j \leqslant k \tag{4}$$

with $1 \leqslant k \leqslant n$.

KP has widely been studied in the literature, see the book of Martello and Toth [7] for a comprehensive illustration of the problem. kKP is the subproblem to be solved when instances of the Cutting Stock Problem with cardinality constraints are tackled by column generation techniques. kKP also appears in processor scheduling problems on computers with k processors and shared memory. Furthermore, kKP could replace KP in the separation of cover inequalities, as outlined in [2].

Throughout this paper, let OPT denote the optimal solution value of the given instance, $w(F) = \sum_{j\in F} w_j$ and $p(F) = \sum_{j\in F} p_j$, where $F \subseteq N$. An algorithm A with solution value z^A is called a $(1-\varepsilon)$-*approximation algorithm*, $\varepsilon \in (0, 1)$, if $z^A \geqslant (1-\varepsilon)OPT$ holds for all problem instances. We will also call $(1-\varepsilon)$ the *approximation ratio* of algorithm A.

Known results: It is well known that KP is NP-hard but pseudopolynomially solvable through dynamic programming, and the same properties hold for kKP [2]. Basically, the developed approximation approaches for KP and kKP can be divided into three groups:

(1) *Approximation algorithms*: For KP the classical $\frac{1}{2}$-approximation algorithm [6] needs only $O(n)$ running time. An approximation ratio of $\frac{1}{2}$ can be obtained also for kKP by rounding the solution of the linear programming relaxation of the problem [2]; this algorithm can be implemented to run in linear time when the LP relaxation of kKP is solved by using the method by Megiddo and Tamir [9].

(2) *Polynomial time approximation schemes* (PTAS): PTAS reach any given approximation ratio and have a running time polynomial in the length of the encoded input. The best schemes currently known requiring linear space are given in Caprara et al. [2]: they yield an approximation ratio of $(1-\varepsilon)$ within $O(n^{\lceil 1/\varepsilon\rceil-2} + n\log n)$ and $O(n^{\lceil 1/\varepsilon\rceil-1})$ running time, for KP and kKP, respectively.

(3) *Fully polynomial time approximation schemes* (FPTAS): FPTAS also reach any given approximation ratio and have a running time polynomial in the length of the encoded input and in the reciprocal of the approximation ratio. This improvement compared to (1) and (2) is usually paid off by larger space requirements, which increases rapidly with the accuracy ε. The first FPTAS for KP was proposed by Ibarra and Kim [4], later on improved by Lawler [6] and Kellerer and Pferschy [5]. In Caprara et al. [2] it is shown that kKP admits an FPTAS that runs in $O(nk^2/\varepsilon)$ time and requires space $O(n + k^3/\varepsilon)$.

New results: Rounding the input is a widely used technique to obtain polynomial time approximation schemes [3,10]. *Arithmetic or geometric rounding* are the most successfully and broadly used ways of rounding to obtain a simpler instance that may be solved in polynomial time (see Section 3 for an application of these to kKP). We contribute by presenting a new technical idea. We show that an appropriate combination of arithmetic and geometric rounding techniques yields a novel and improved rounding method. To the best of our knowledge, these techniques have never been combined together before. Moreover, we present a PTAS for kKP requiring linear space and running time $O(n + k(\log 1/\varepsilon)^{O(1/\varepsilon)})$. Our algorithm is clearly superior to the one in [2], and it is worth noting that the running time contains no exponent on n dependent on ε. Since KP is a special case of kKP, we also speed up the previous result for KP to $O(n(\log 1/\varepsilon)^{O(1/\varepsilon)})$. Finally, denote $z = \min\{k, 1/\varepsilon\}$, we present a faster FPTAS for kKP that runs in $O(n + kz^2/\varepsilon^2)$ time and has a bound of $O(n + z^3/\varepsilon)$ on space requirements.

2. The main ideas of the paper

Let $p_{\max} = \max_j p_j$ be the maximum item profit, and let P^H denote the solution value obtained in $O(n)$ time for a given instance I of kKP by employing the $\frac{1}{2}$-approximation algorithm $H^{1/2}$ described in [2]. As shown in [2],

$$P^H \leqslant OPT \leqslant P^H + p_{\max} \leqslant 2P^H. \tag{5}$$

The construction of our approximation schemes is based on the implementation of two ideas. The first idea is quite standard for every approximation scheme. It consists in dividing the whole set of items (of instance I) into two subsets of "large" and "small" items, where the "size" of an item j is determined by comparing its profit p_j with a properly specified threshold x^*. Specifically, items with profits $p_j \geqslant x^*$ are called *large*, while the remaining ones are *small*; x^* is chosen so that the number of large items is bounded by an amount dependent on ε. To the set of large items ($\mathscr{L}$) an enumerative procedure is applied which can guarantee that the optimal subset of large items (that the optimal solution contains) is not missed. At the same time, to the set of small items ($\mathscr{S}$) we apply the above mentioned efficient approximation algorithm $H^{1/2}$ which guarantees an absolute error not greater than $\max_{j\in\mathscr{S}} p_j \leqslant x^*$.

The second idea is the application of a (linear time) *rounding-and-reduction* procedure to a given set of items N prior to implementing the approximation scheme itself. The goal of that procedure consists in a drastic reduction of the number of items $n = |N|$ to an amount independent of n (dependent only on ε and k). This enables one to reduce the running time of any $(1-\varepsilon)$-approximation algorithm $\mathscr{A}_\varepsilon$ to $\mathrm{O}(n + k\cdot\varphi(\varepsilon))$ with required space also about $\mathrm{O}(n+\psi(\varepsilon))$, where the functions $\varphi(\varepsilon)$ and $\psi(\varepsilon)$ are specific for a particular approximation scheme (be it a PTAS or an FPTAS).

The idea of the rounding-and-reduction procedure is based on the following observation. If an instance under consideration contains several items with the same profit p_j, then any optimal solution of the kKP may contain no more than

$$\min\left\{k, \left\lfloor \frac{OPT}{p_j} \right\rfloor\right\} \leqslant \min\left\{k, \left\lfloor \frac{2P^H}{p_j} \right\rfloor\right\} \doteq \lambda(p_j)$$

items of profit p_j. Therefore, given an instance I, we can sort the items by their profits, and for each particular profit p_j retain no more than $\lambda(p_j)$ items (if any) with smallest weights—without affecting the optimum. This procedure is called a *Reduction* of the set of items.

The above reduction procedure may happen to be useless if all profits turn to be different. To avoid this trouble, we first implement a *rounding* procedure. According to the latter, a discrete grid $S = \{y_1, y_2, \ldots, y_\beta\}$ $(0 \leqslant y_1 < y_2 < \cdots < y_\beta \leqslant p_{\max})$ is defined in the range of profit values (the number β of nodes of grid S being independent of the number of items n), and then each profit $p_j \geqslant y_1$ is rounded down to the nearest grid node y_i. Applying then the reduction procedure, we come to a reduced set (N_1) of items with profits $p_j \geqslant y_1$ (we will show later how to choose y_1 in order that the whole set of items N could be put on reduction). Its cardinality can be estimated as

$$|N_1| \leqslant \sum_{i=1}^{\beta} \lambda(y_i). \tag{6}$$

3. Rounding and reduction

Unlike the reduction procedure, the rounding procedure evidently may affect the optimum, because it decreases the profits of items. To estimate this loss of optimality, two techniques can be implemented: estimating an absolute loss or a relative loss.

When we estimate an absolute loss, we observe that a profit p_j lying in the interval $[y_i, y_{i+1})$ may lose no more than $(y_{i+1} - y_i)$. Keeping in mind that the optimal solution may contain no more than $\lambda(y_1)$ items with profits $p_j \geqslant y_1$, we can estimate the total loss of optimality (caused by the rounding procedure) by the amount $\lambda(y_1)\cdot\max_i\{y_{i+1} - y_i\}$. Specifically, if grid S represents an arithmetical progression $S_{\mathrm{a}}(y_1, d)$ (with a constant difference $y_{i+1} - y_i = d$ and the first point y_1), the loss is bounded by $\lambda(y_1)d$. Choosing $d = \delta P^H/\lambda(y_1)$, by (5) we can derive that the relative error is not greater than δ—a rounding procedure which employs such a technique of estimating the loss of optimality (with an arithmetical grid) will be referred to as an *arithmetical rounding*.

We come to another technique of estimating the error, when we observe that the profit of each item can be decreased (in the worst case) by a factor of $z \doteq \max_i y_{i+1}/y_i$—and so does the total profit of the optimal solution, no matter how many items are contained in that solution. Thus, the relative error can be estimated as $(OPT - OPT')/OPT = 1 - OPT'/OPT \leqslant 1 - 1/z$, where OPT' is the optimal profit of the instance after the above rounding is applied. In particular, if grid S represents a geometric series $S_{\mathrm{g}}(y_1, \delta)$ (with the first node at y_1 and a constant ratio $y_{i+1}/y_i \equiv z = 1/(1-\delta)$), then the relative error can be bounded above by δ. If a rounding procedure employs this method of estimating the error

(with a geometric grid), it will be referred to as a *geometric rounding*. Notice that the relative error obtained for the geometric grid $S_g(y_1, \delta)$ does not depend on the choice of the first node y_1.

Normally, one of the above two rounding techniques (arithmetical or geometric) is used within constructions of approximation schemes for kKP. In our paper we demonstrate that applying a proper combination of these techniques provides a stronger reduction of the set of items. (The most significant improvement can be obtained for FPTAS.) Let us try to understand why this happens.

Firstly, in order that the whole set of items N could be put on reduction, we should choose $y_1 = 0$. Since no node of the geometric grid can be put to zero, y_1 should be a node of an arithmetical grid. Thus, to provide the δ relative loss under the arithmetical reduction, we should put $d = \delta P^H/k$. It can be observed from (6) that the sparser is the grid $\{y_i\}$ in the range of profit values (and the wider are the steps of the grid), the stronger is the reduction of the item set. While in an arithmetical grid $S_a(a_1, d) = \{a_1, a_2, \ldots\}$ all steps are of a constant length $a_{i+1} - a_i = d$, this is not the case for a geometric grid. While being very small in the vicinity of zero, they become wider and wider, as the grid nodes are moving off zero. Specifically, in a grid $S_g(g_1, \delta) = \{g_1, g_2, \ldots\}$ (that provides the relative loss δ) the step that directly precedes the node g_i has length δg_i. Therefore, choosing the first node of the grid $S_g(g_1, \delta)$ at point g_1 such that $\delta g_1 = d = \delta P^H/k$, we obtain a grid having all steps wider than d.

Thus, the best strategy for the item reduction is implementing a *hybrid rounding* technique: in the range of profit values $[0, P^H/k)$ an arithmetical rounding (with grid $S_a(0, \delta P^H/k)$) should be applied; it should be changed to a geometric rounding (with grid $S_g(P^H/k, \delta)$) in the range $[P^H/k, P^H]$.

Yet in order to simplify further calculation, we will implement the hybrid rounding with the *toggle point* $2P^H/k$ (instead of P^H/k), so that it coincides with the point where the function $\lambda(x)$ toggles its definition. This will not produce much extra loss in reduction (none in the order of magnitude), but allows us to avoid considering (in approximation schemes) the additional case of x^* between P^H/k and $2P^H/k$.

Let us estimate the cardinality of the reduced set of items N_1 obtained as a result of the *hybrid rounding-and-reduction* procedure (*HRR*, for short) with a hybrid grid $S_{ag} = \{y_1, y_2, \ldots\}$ coinciding with the arithmetical grid $S_a(0, \delta P^H/k)$ for $y_i < 2P^H/k$, and with the geometric grid $S_g(2P^H/k, \delta)$ for $y_i \geqslant 2P^H/k$. Discarding the items with profits $p_j = y_1 = 0$, we can derive

$$\left|\left\{j \in N_1 \mid p_j < \frac{2P^H}{k}\right\}\right| \leqslant \sum_{0<y_i<2P^H/k} \lambda(y_i) \leqslant k\left\lfloor \frac{2P^H}{kd}\right\rfloor = k\left\lfloor \frac{2}{\delta}\right\rfloor \leqslant \frac{2k}{\delta}.$$

Let $y_{i^*} = 2P^H/k$ be the first node of the geometric part of the hybrid grid. Then for the remaining items of N_1 we have the following inequalities:

$$\left|\left\{j \in N_1 \mid p_j \geqslant \frac{2P^H}{k}\right\}\right| \leqslant \sum_{y_i \geqslant 2P^H/k} \lambda(y_i) \leqslant \sum_i \frac{2P^H}{y_i} \leqslant \frac{2P^H}{y_{i^*}} \sum_{i=0}^{\infty}(1-\delta)^i = \frac{2P^H}{y_{i^*}\delta} = \frac{k}{\delta}.$$

Thus, we can derive the following bound on the cardinality of the reduced set:

$$|N_1| \leqslant \frac{3k}{\delta}. \tag{7}$$

We can also estimate the number β of nodes of the hybrid grid in the range of profit values. As established above, the number of nodes in the "arithmetical" range (from 0 to y_{i^*}) is not greater than $2/\delta$. The number of nodes in the geometric part of the grid (from y_{i^*} to $p_{\max}$) is equal to the maximum Δ such that $y_{i^*+\Delta-1} = y_{i^*}/(1-\delta)^{\Delta-1} \leqslant p_{\max} \leqslant P^H$. Since $y_{i^*} = 2P^H/k$, we can derive the bound

$$\Delta \leqslant 1 + \left\lfloor \frac{\ln k/2}{\ln 1/(1-\delta)}\right\rfloor = \mathrm{O}\left(\frac{1}{\delta}\log k\right).$$

Summing up the two bounds, we obtain

$$\beta = \mathrm{O}\left(\frac{1}{\delta}\log k\right). \tag{8}$$

The relative loss of optimality caused by this hybrid rounding is clearly bounded by 2δ. Finally, the running time of the HRR procedure can be estimated as $\mathrm{O}(n + (1/\delta)\log k)$, and the same amount estimates the required space. However, since one of the goals of the paper is to obtain a PTAS requiring linear space, we observe that the HRR procedure can be implemented to run in $\mathrm{O}(n + \beta \log \beta)$ time and $\mathrm{O}(n)$ space, where the hidden constant in the space requirement is independent on δ. Indeed, first round profits as described. This task can be done in $\mathrm{O}(n)$ time. The next step consists of partitioning the n items into subsets of items with the same rounded profits. If $n \geqslant \beta$, this task can be easily done in $\mathrm{O}(n)$ time and space. Otherwise, if $n < \beta$, first sort the items according to their profit values, and then partition the items into at most β subsets. The latter is a task that can be done in $\mathrm{O}(\beta \log \beta)$ time and $\mathrm{O}(n)$ space. Finally, the last step consists of reducing the item set. Let $\mathscr{P}_i$ denote the set of items with the same rounded profit value p_i. The subsequent reduction of each set $\mathscr{P}_i$ (so as to leave at most $\lambda(p_i)$ items with the least weights) can be done in time linear in $|\mathscr{P}_i|$ (due to the result from [1]). Therefore, in total (over all sets $\mathscr{P}_i$) the time for selecting the items with the smallest weights is linear. Thus, we come to the following result.

Lemma 1. *Given an instance of the kKP with n items and a positive* $\delta \in (0, \frac{1}{2})$, *the number of items can be reduced* (*by the HRR procedure*) *down to* $3k/\delta$ *items*, *with at most* 2δ *relative loss in optimal profit*. *The procedure can be implemented to run in* $\mathrm{O}(n + ((1/\delta)\log k)\log((1/\delta)\log k))$ *time and linear* (*in n*) *space requirement*.

Remark. It is easily understood that using a geometric rounding-and-reduction only (with grid $S_{\mathrm{g}}(\delta P^H/k, \delta)$ and with discarding the items with profits $p_j \leqslant \delta P^H/k$) leads to a weaker reduction, because of a large number of "short" steps $(y_{i+1} - y_i)$ in the first part of the grid (for $y_i < 2P^H/k$). Alternatively, using an arithmetical rounding only (with grid $S_{\mathrm{a}}(0, \delta P^H/k)$) also leads to a weaker reduction, because of a large number of short steps in the second part of the grid (in comparison with geometric steps in this range of profits). The reader can easily check these statements.

4. An improved PTAS for *k*KP

Suppose we are given a set of items N_1 reduced with respect to a grid $S = \{y_1, y_2, \ldots\}$. The latter means that each item profit p_j $(j \in N_1)$ lying in the interval $[y_1, \infty)$ coincides with one of the grid nodes $\{y_i\}$, and that $|\mathscr{P}_i| \leqslant \lambda(y_i)$ for each $\mathscr{P}_i \doteq \{j \in N_1 \mid p_j = y_i\}$ $(i = 1, 2, \ldots)$. The construction of the PTAS described below is based on dividing the set of items N_1 into two subsets according to their profits being compared with a given threshold x^*: items from $\mathscr{S} \doteq \{j \in N_1 \mid p_j < x^*\}$ are called *small*, and those from $\mathscr{L} \doteq \{i \in N_1 \mid p_i \geqslant x^*\}$ are called *large*.

Next we define the notion of a "configuration". Let $\{y'_1, y'_2, \ldots, y'_{\beta'}\}$ be the set of all "large" nodes of grid S: $x^* \leqslant y'_1 < y'_2 < \cdots < y'_{\beta'} \leqslant P^H$, and let $T \subseteq \mathscr{L}$ be a subset of large items. The vector $\eta(T) \doteq (\eta_1(T), \ldots, \eta_{\beta'}(T))$ with $\eta_i(T) = |T \cap \mathscr{P}_i|$ $(i = 1, \ldots, \beta')$ is called a *configuration* of set T. An arbitrary vector $\bar{\eta} = (\bar{\eta}_1, \ldots, \bar{\eta}_{\beta'}) \in \mathbb{N}^{\beta'}$ will be referred to as a *feasible configuration*, if $\bar{\eta}_i \leqslant \eta_i(\mathscr{L})$ $(i = 1, \ldots, \beta')$ and $\sum_{i=1}^{\beta'} \bar{\eta}_i \leqslant \bar{\lambda} \doteq \lambda(x^*)$.

For any $\varepsilon \in (0, 1)$ we present a $(1 - \varepsilon)$-approximation algorithm $\mathscr{A}_\varepsilon$ consisting of 5 steps.

Algorithm $\mathscr{A}_\varepsilon$

(**S1**) Compute the profit value P^H returned by the $H^{1/2}$ approximation algorithm applied to the kKP problem with the initial set of items N. Initialize the solution A with the solution value $P^A = 0$.

(**S2**) Choose a real value $\delta = C\varepsilon$ for some $C \in (0, 1/2)$ and apply the HRR procedure with grid $S_{\mathrm{ag}} = \{y_1, y_2, \ldots\}$ (described in the previous section) to obtain a reduced set of items $N_1 \subseteq N$ with at most $3k/\delta$ items and 2δ relative loss in profit (as shown in Lemma 1).

(**S3**) Put $x^* = (\varepsilon - 2\delta)P^H$ to be a threshold between the profit values of *small* and *large* items: $\mathscr{S} \doteq \{i \in N_1 \mid p_i < x^*\}$, $\mathscr{L} \doteq \{i \in N_1 \mid p_i \geqslant x^*\}$. Number the large items in each set $\mathscr{P}_i \doteq \{j \in N_1 \mid p_j = y_i\}$ $(y_i \geqslant x^*)$ in nondecreasing order of their weights.

(**S4**) **For each** feasible configuration $\bar{\eta} = (\bar{\eta}_1, \ldots, \bar{\eta}_{\beta'})$ (defined with respect to grid S_{ag}, threshold x^* and set $\mathscr{L}$) **do begin**

From among the set of subsets of large items with configuration $\bar{\eta}$ choose the one with the least weight: $L' := \arg\min_{\{L \subseteq \mathscr{L} \mid \eta(L) = \bar{\eta}\}} w(L)$—Clearly, to obtain the desired subset, it suffices in each list $\mathscr{P}_{\beta - \beta' + i}$ $(i = 1, \ldots, \beta')$ to take the first $\bar{\eta}_i$ items. (Here β' is the number of "large" nodes and β is the overall number of nodes of grid S_{ag} in the range of item profits.)

If $w(L') > c$, pass on to considering the next configuration $\bar{\eta}$. Otherwise, on the set of small items $\mathscr{S}$ consider the k'KP problem $S(L')$ with $k' = k - |L'|$ and the knapsack capacity $c' = c - w(L')$. Applying the algorithm $H^{1/2}$ to problem $S(L')$, obtain an approximate solution $S' \subseteq \mathscr{S}$ with profit value

$$p(S') \geqslant OPT(S(L')) - \max_{j \in \mathscr{S}} p_j \geqslant OPT(S(L')) - x^*. \tag{9}$$

If $p(S') + p(L') > P^A$, set $A := S' \cup L'$, $P^A := p(S') + p(L')$.

end

(**S5**) Return the solution A.

Theorem 1. *Given $\varepsilon \in (0, 1)$, algorithm $\mathscr{A}_\varepsilon$ provides a $(1 - \varepsilon)$-approximation for any instance of the kKP problem. It runs in time $\mathrm{O}(n + k(\log 1/\varepsilon)^{\mathrm{O}(1/\varepsilon)})$ and requires linear space.*

Proof. First, by Lemma 1 we have

$$OPT(N_1) \geqslant (1 - 2\delta)OPT. \tag{10}$$

Let T^* denote the optimal solution to the kKP with the item set N_1; $L^* \doteq T^* \cap \mathscr{L}$, $S^* \doteq T^* \cap \mathscr{S}$. It is clear that S^* is the optimal solution to the problem $S(L^*)$ specified for the item set $\mathscr{S}$ with the capacity constraint $c - w(L^*)$ and with at most $k - |L^*|$ items. Thus, we have

$$OPT(N_1) = p(T^*) = p(L^*) + OPT(S(L^*)). \tag{11}$$

Let L' be the subset of large items chosen at the iteration of step (S4) at which the configuration $\bar{\eta} = \eta(L^*)$ was considered. Since L' has the same configuration as L^*, it has also the same profit and the same cardinality; furthermore, it has a less weight: $w(L') \leqslant w(L^*)$. Thus, we may conclude that the generated problem for small items $(S(L'))$ has weaker constraints as compared with problem $S(L^*)$, and so, we have

$$OPT(S(L')) \geqslant OPT(S(L^*)). \tag{12}$$

Let S' be the item set returned at that iteration as a solution to the $S(L')$ problem. Then by (9)–(12) we can derive

$$\begin{aligned} P^A &\geqslant p(S') + p(L') \geqslant OPT(S(L')) - x^* + p(L^*) \\ &\geqslant OPT(S(L^*)) + p(L^*) - x^* = OPT(N_1) - x^* \\ &\geqslant (1 - 2\delta)OPT - (\varepsilon - 2\delta)OPT = (1 - \varepsilon)OPT. \end{aligned}$$

In order to derive the bound on running time, we first need to estimate some amounts in terms of ε.

First of all, putting $\delta = C\varepsilon$ (for a constant $C \in (0, 1/2)$), we obtain relations

$$\frac{1}{\delta} = \mathrm{O}\left(\frac{1}{\varepsilon}\right), \quad \frac{1}{\varepsilon - 2\delta} = \mathrm{O}\left(\frac{1}{\varepsilon}\right) \tag{13}$$

by means of which we can derive the bound on $\bar{\lambda}$:

$$\bar{\lambda} \leqslant \frac{2P^H}{x^*} = \frac{2}{\varepsilon - 2\delta} = \Theta\left(\frac{1}{\varepsilon}\right). \tag{14}$$

Finally, let α be the number of large items. We observe that

$$\beta' = \mathrm{O}\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right), \quad \alpha = \mathrm{O}\left(\frac{1}{\varepsilon^2}\right). \tag{15}$$

Indeed, if $k \leqslant 2/(\varepsilon - 2\delta)$ then the claim follows by (7), (8) and (13); otherwise, we have $(\varepsilon - 2\delta)P^H > 2P^H/k$ and large profits are bigger than the toggle point $2P^H/k$, which means that the nodes $\{y'_1, y'_2, \ldots, y'_{\beta'}\}$ are nodes of a geometrical grid having $y'_1 \geqslant (\varepsilon - 2\delta)P^H$, which easily implies the claimed bound on α and β' by arguments similar to those used in Section 3.

Now we return to deriving the bounds on running time and space. As announced in [2], step (S1) can be implemented in $\mathrm{O}(n)$ time. At step (S2), by Lemma 1 and (13), the HRR procedure can run in $\mathrm{O}(n + (1/\varepsilon)\log k)\log((1/\varepsilon)\log k))$ time and linear space requirement. The time complexity of step (S3) is determined by the time to order large items in each set $\mathscr{P}_i$, which is bounded by $\mathrm{O}(\alpha \log \alpha)$, therefore $\mathrm{O}((1/\varepsilon^2)\log 1/\varepsilon)$ time by using (15). The most demanding part is step (S4). This step requires to run algorithm $H^{1/2}$, which takes $\mathrm{O}(|\mathscr{S}|) = \mathrm{O}(k/\varepsilon)$ time, for each feasible configuration. So the time complexity of step (S4) is completely determined by upper bounding the number of feasible configurations, which are at most the number of configurations with no more than $\bar{\lambda}$ items, i.e., $\binom{\beta'+\bar{\lambda}}{\bar{\lambda}} = \mathrm{O}((\beta'/\bar{\lambda})^{2\bar{\lambda}}) = (\log 1/\varepsilon)^{\mathrm{O}(1/\varepsilon)}$. □

To compare our algorithm with the one provided in [2] notice that the running time of the latter is $\mathrm{O}(n^{\lceil 1/\varepsilon\rceil - 1})$, whereas our scheme is linear. As in [2], our algorithm can be easily modified to deal with the *Exact k-item Knapsack Problem*, that is a kKP in which the number of items in a feasible solution must be exactly equal to k. The time and space complexities, and the analysis of the resulting algorithm are essentially the same as the one described above.

5. An improved FPTAS for kKP

In this section we propose a fully polynomial time approximation scheme. This FPTAS is obtained by replacing step (S4) of algorithm $\mathscr{A}_\varepsilon$ (see Section 4) with a new one (S4′), and by setting $x^* = (\varepsilon - 3\delta)P^H$ at step (S3). The resulting modification of algorithm $\mathscr{A}_\varepsilon$ will be referred to as $\mathscr{A}'_\varepsilon$. One of the problems with algorithm $\mathscr{A}_\varepsilon$ is that constructing all possible configurations at step (S4) requires considering an exponential in $1/\varepsilon$ number of cases. To avoid this exponential dependence (our aim is to obtain a fully polynomial approximation scheme), we use for large items the dynamic programming algorithm proposed in [2] and a hybrid rounding-and-reduction procedure.

Description of the new step (*S4′*): Let $\bar{\lambda} = \lambda(x^*)$ denote an upper bound on the maximum number of large items (i.e., those with profits $p_j \geqslant x^*$) that can be simultaneously used in a feasible solution. Step (S4′) starts applying an arithmetical rounding, with grid $S_{\mathrm{a}}(x^*, \delta P^H/\bar{\lambda})$, to the set of large items. Let $\mathscr{L}_a$ be this new, arithmetically rounded, set of large items; $\alpha = |\mathscr{L}_a|$. For simplicity of notation, renumber the set of items so that the first α items were large. By G we denote the set of values of the arithmetical grid $S_{\mathrm{a}}(x^*, \delta P^H/\bar{\lambda})$ in the interval $[x^*, 2P^H]$ plus value zero. Consider any subset of large items that is also a feasible solution for the kKP problem, then observe that their total profit sum is a value that belongs to G. Let $\gamma = |G|$ be the number of different values in G.

Compute all the subsets of large items with different profit sums, different values of cardinality and the minimum weight. This can be done efficiently by using a new form of dynamic program (DP) as explained in [2] (see also Realization A in Section 5.1). We describe DP in the following for the sake of completeness.

Denote by $g_i(a, \ell)$, for $i = 1, \ldots, \alpha$, $a \in G$, $\ell = 0, 1, \ldots, \bar{\lambda}$, the optimal solution of the following problem:

$$g_i(a,\ell) = \min\left\{ \sum_{j=1}^{i} w_j x_j \;\middle|\; \begin{array}{l} \sum_{j=1}^{i} p_j x_j = a; \\ \sum_{j=1}^{i} x_j = \ell; \\ x_j \in \{0,1\}, \;\; j = 1, \ldots, i \end{array} \right\}.$$

One initially sets $g_0(a, \ell) = +\infty$ for all $\ell = 0, \ldots, \bar{\lambda}$, $a \in G$, and then $g_0(0, 0) = 0$. Then, for $i = 1, \ldots, \alpha$ the entries for g_i can be computed from those of g_{i-1} by using the formula

$$g_i(a,\ell) = \min\left\{ \begin{array}{l} g_{i-1}(a,\ell), \\ g_{i-1}(a - p_i, \ell - 1) + w_i \quad \text{if } \ell > 0 \text{ and } a \geqslant p_i \end{array} \right\}. \tag{16}$$

In Section 5.1 we will describe how to compute efficiently a *solution* (i.e., a subset of large items $L(a,\ell) \subseteq \{1,\dots,\alpha\}$ such that $\sum_{j\in L(a,\ell)} p_j = a$, $\sum_{j\in L(a,\ell)} w_j = g_\alpha(a,\ell)$ and $|L(a,\ell)| = \ell$) for any pair (a,ℓ) with $g_\alpha(a,\ell) < +\infty$.

After the calculation of the function $g_\alpha(a,\ell)$ is completed for all pairs (a,ℓ), if $g_\alpha(a,\ell) \leqslant c$ then we complete solution $L(a,\ell)$ by using small items $\mathscr{S}$, as at step (S4) of algorithm $\mathscr{A}_\varepsilon$ (see Section 4). Namely, we consider problem k'KP(a,ℓ) with $k' = k - \ell$ and the knapsack capacity $c' = c - g_\alpha(a,\ell)$. Apply algorithm $H^{1/2}$ to problem k'KP(a,ℓ) and obtain an approximate solution $S(a,\ell) \subseteq \mathscr{S}$ with profit value $p(S(a,\ell))$. Over all pairs (a,ℓ) with $g_\alpha(a,\ell) \leqslant c$, choose the one that delivers the maximum profit value $\max_{\{(a,\ell)\}}\{p(S(a,\ell)) + a\}$.

5.1. Realization of DP

We describe two principally different realizations of DP. The first realization requires less time but more space than the second. The following descriptions are due to Sevastianov [11].

Realization A [*one forward and one reverse trace*]: The DP-scheme consists of a *forward step* and a *backward step*. The forward step consists of iterations $j = 1,\dots,\alpha$ at which we consecutively compute the functions $g_j(a,\ell)$ for all pairs (a,ℓ). Next we find the optimal pair (a^*,ℓ^*). The backward step is aimed at the restoration of the whole solution $L(a^*,\ell^*)$ for the optimal pair (a^*,ℓ^*) of the last function $g_\alpha(a,\ell)$.

During the forward step we create a tree-like information structure T which proves to be convenient (and efficient) for keeping and extracting the information about the optimal solution $L(a,\ell)$ for all realizable pairs (a,ℓ). So, we construct a directed tree $T = (V,U)$ in which each node $v \in V$ will be associated with an event of refreshing the value of $g(a,\ell)$ (and the solution $L(a,\ell)$) for some pair $(a(v),\ell(v))$ during the iterations of the forward step. (Each such event is accompanied by arising a new node in T.) Let $v(a,\ell)$ denote the node in T associated with the event of the last refreshment of the solution $L(a,\ell)$. Thus, the value of $v(a,\ell)$ for a given pair (a,ℓ) may change during the forward step of DP. Meanwhile, the pair $(a(v),\ell(v))$ for a given node $v \in V$ remains permanent during the whole life period of this node.

For each node $v \in V$ we store:

- item $j(v)$ last added to the solution $L(a(v),\ell(v))$ at the moment of arising the node v in T;
- node $p(v)$ being "parent" to node v in the rooted tree T;
- counter $c(v)$ of pairs (a,ℓ) for which event v is currently actual (when $c(v)$ becomes equal to zero, node v is deleted from T).

Now let us present a formal description of the DP-scheme.

Forward step.
Initialization:
for $a \in G$ **do for** $\ell = 0,\dots,\bar{\lambda}$ **do** $\{v(a,\ell) := \mathbf{nul};\ g(a,\ell) := \infty\}$;
$V := \{v_0\};\ v(0,0) := v_0;\ g(0,0) := 0;\ c(v_0) := 1;\ p(v_0) := \mathbf{nul};\ j(v_0) := \mathbf{nul}$;
Iterations:
for $j := 1,\dots,\alpha$ **do**
  for $\ell := \bar{\lambda}$ **down to** 1 **do** % this **decreasing** order for ℓ is essential
    for $a \in G$ **do if** $(a \geqslant p_j)\&(\ell > 0)$ **then begin**
      $t := g(a - p_j, \ell - 1) + w_j$;
      if $t < g(a,\ell)$ **then begin**
        $g(a,\ell) := t$; % $g(a,\ell)$ updates its value
        $x := v(a,\ell)$;
      M : $c(x) := c(x) - 1$; **if** $c(x) = 0$
        then $\{V := V\backslash\{x\};\ x := p(x);$ **goto** $M\}$
        % Generate a new node b and add it to V:
        $V := V \cup \{b\};\ j(b) := j;\ c(b) := 1$;
        $y := v(a - p_j, \ell - 1);\ p(b) := y;\ c(y) := c(y) + 1$
      end {if}
    end {for}
Find the optimal pair (a^*,ℓ^*).

Backward step (restoring the solution L for the optimal pair (a^*, ℓ^*)).
$L := \emptyset$; $x := v(a^*, \ell^*)$;
while $p(x) \neq$ **nul** **do** $\{L := L \cup \{j(x)\}$; $x := p(x)\}$.
Clearly, each of the $O(\gamma\bar{\lambda})$ leaves of the tree T has at most $\bar{\lambda}$ ancestors (the nodes from $v(a, \ell)$ backward to the root of T) which yields the $O(\gamma\bar{\lambda}^2)$ space bound. To get the time complexity of $O(\gamma\bar{\lambda}\alpha)$, it suffices to observe that

(1) there can be no more than $\gamma\bar{\lambda}\alpha$ updates of function $g(a, \ell)$ (and so, no more than this number of nodes can be added to the tree T) during the whole forward step;
(2) each update is accompanied by at most a constant number of elementary calculations within the loop "if ... end {if}"—not counting the inner loops on label M;
(3) each iteration of the loop on M is accompanied by deleting a node from T; since the number of deleted nodes cannot exceed the total number of added nodes, the total number of loops on M cannot exceed $\gamma\bar{\lambda}\alpha$;
(4) each iteration of the loop on M requires constant time.

Realization B [*iteratively reconstructed solution*]: For each pair (a, ℓ) at the forward trace of DP we memorize only the item $v(a, \ell)$ last added to the solution $L(a, \ell)$. To reconstruct the complete solution $L(a^*, \ell^*)$ for the optimal pair (a^*, ℓ^*), we subtract the item $i = v(a^*, \ell^*)$ from the optimal solution $L(a^*, \ell^*)$, deriving that the pair $(a', \ell') = (a^* - p_i, \ell^* - 1)$ should be optimal for a subproblem with a shorter set of items $\{1, \ldots, i-1\}$ (and reduced knapsack capacity and cardinality). Yet since $v(a', \ell')$ may change up to the end of the forward trace, to restore its true value, we have to repeat the forward trace (this time, for a smaller set of items $\{1, \ldots, i-1\}$), and so forth. As a result, to restore the whole optimal solution, we will have to perform the forward trace about $O(\bar{\lambda})$ times.

Analysis: The solution value returned by the described algorithm is within $(1-\varepsilon)$ times the optimal value. This claim can be easily verified by using similar arguments as in the proof of Theorem 1, where the only difference is that we have to take into consideration the error made by the additional arithmetic rounding performed at the beginning of the new step (S4′).

After the calculation of function $g_\alpha(a, \ell)$ is completed for all pairs (a, ℓ), the computation of $\max_{\{(a,\ell)\}}\{p(S(a, \ell))+a\}$ requires time $O(s\gamma\bar{\lambda})$, where s is the number of small items. Thus, for scheme A, the overall running time can be bounded by $O(n+\gamma\bar{\lambda}(s+\alpha))$, while the space requirement is $O(n+\gamma\bar{\lambda}^2)$. On the other hand, scheme B requires $O(n+\gamma\bar{\lambda}(s+\alpha\bar{\lambda}))$ time and $O(n+\gamma\bar{\lambda})$ space. Denote $z = \min\{k, 1/\varepsilon\}$. Then we have

$$\bar{\lambda} \leqslant O(z), \quad \alpha \leqslant O\left(\frac{z}{\varepsilon}\right), \quad \gamma \leqslant O\left(\frac{\bar{\lambda}}{\varepsilon}\right) \leqslant O\left(\frac{z}{\varepsilon}\right), \quad s \leqslant O\left(\frac{k}{\varepsilon}\right).$$

For the different DP-schemes we obtain bounds on running time and space requirement in the form of $O(n+\varphi(k, \varepsilon, z))$ and $O(n+\psi(k, \varepsilon, z))$, respectively. Specifically, we have

Scheme	$\varphi(k, \varepsilon, z)$ (running time)	$\psi(k, \varepsilon, z)$ (space requirement)
A	$O(\gamma\bar{\lambda}s) \leqslant O\left(\frac{kz^2}{\varepsilon^2}\right)$	$O(\gamma\bar{\lambda}^2) \leqslant O\left(\frac{z^3}{\varepsilon}\right)$
B	$O(\gamma\bar{\lambda}s + \alpha\gamma\bar{\lambda}^2) \leqslant O\left(\frac{kz^2+z^4}{\varepsilon^2}\right)$	$O(\gamma\bar{\lambda}) \leqslant O\left(\frac{z^2}{\varepsilon}\right)$

Theorem 2. *Given* $\varepsilon \in (0, 1)$, *algorithm* $\mathscr{A}'_\varepsilon$ *provides a* $(1-\varepsilon)$*-approximation for any instance of the kKP problem. It runs in time* $O(n+\varphi(k, \varepsilon, z))$ *and requires space* $O(n+\psi(k, \varepsilon, z))$, *where* $z = \min\{k, 1/\varepsilon\}$ *and functions* φ *and* ψ *depend on a concrete realization of the dynamic programming scheme used within* $\mathscr{A}'_\varepsilon$. *Specifically, for scheme A we have* $\varphi(k, \varepsilon, z) \leqslant O(kz^2/\varepsilon^2)$ *and* $\psi(k, \varepsilon, z) \leqslant O(z^3/\varepsilon)$; *alternatively, for scheme B we have* $\varphi(k, \varepsilon, z) \leqslant O((kz^2+z^4)/\varepsilon^2)$ *and* $\psi(k, \varepsilon, z) \leqslant O(z^2/\varepsilon)$.

Remark. The complexity of the described FPTAS depends on the number of large and small items, and on the size γ' of the set of all realizable total profit values of large items. We have already noted (see Lemma 1) that the number of items (and therefore also the number of small items) can be reduced by using the HRR procedure, i.e., a combination of the arithmetic and geometric rounding-and-reduction procedures. By the arguments of Section 3, it can be promptly

checked that the number of large items can be bounded by $O(1/\varepsilon^2)$ by using only the geometric rounding-and-reduction procedure. However, the value of γ' can be bounded above by $\gamma \leqslant O(z/\varepsilon)$ only after the large items are subject to the additional arithmetical rounding (as we did at the beginning of step (S4′)). Otherwise, as shown in [8], we have Claim 1.

Claim 1. *The number of realizable total profit values γ' can be exponential in $1/\varepsilon$, if a geometric rounding only is applied to large items.*

This again demonstrates the advantage of the hybrid rounding technique.

Acknowledgements

We are profoundly grateful to Sergey Sevastianov who pointed out some mistakes and helped us to improve the readability of this paper considerably. Besides, he suggested the improvement to the running time of our PTAS from $O(n + k(1/\varepsilon)^{O(1/\varepsilon)})$ to $O(n + k(\log 1/\varepsilon)^{O(1/\varepsilon)})$. The present work deeply profited by his patient and accurate work. Thanks are due to Klaus Jansen for introducing us to the k-item knapsack problem.

Supported by the Swiss National Science Foundation projects 200021-104017/1 "Power Aware Computing", and 200021-100539/1 "Approximation Algorithms for Machine scheduling Through Theory and Experiments", and 2000-61847.00 "Unification of universal inductive inference and sequential decision theory" (to Jürgen Schmidhuber).

References

[1] M. Blum, R. Floyd, V. Pratt, R. Rivest, R. Tarjan, Time bounds for selection, J. Comput. System Sci. 7 (1973) 448–461.
[2] A. Caprara, H. Kellerer, U. Pferschy, D. Pisinger, Approximation algorithms for knapsack problems with cardinality constraints, European J. Oper. Res. 123 (2000) 333–345.
[3] D. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems, PWS Publishing Company, Boston, 1995.
[4] O. Ibarra, C. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, J. ACM 22 (1975) 463–468.
[5] H. Kellerer, U. Pferschy, A new fully polynomial approximation scheme for the knapsack problem, APPROX'98, Lecture Notes on Computer Science 1444 (1998) 123–134.
[6] E. Lawler, Fast approximation algorithms for knapsack problems, in: Proceedings of the 18th Annual Symposium on Foundations of Computer Science, 1977, pp. 206–218.
[7] S. Martello, P. Toth, Knapsack Problems, Wiley, New York, 1990.
[8] M. Mastrolilli, M. Hutter, Hybrid rounding techniques for knapsack problems, Technical Report IDSIA-03-02, Manno-Lugano, CH, 2002. http://arxiv.org/abs/cs.CC/0305002
[9] N. Megiddo, A. Tamir, Linear time algorithms for some separable quadratic programming problems, Oper. Res. Lett. 13 (1993) 203–211.
[10] P. Schuurman, G. Woeginger, Approximation schemes—a tutorial, in: R. Moehring, C. Potts, A. Schulz, G. Woeginger, L. Wolsey (Eds.), Lectures on Scheduling, 2001, to appear. http://wwwhome.cs.utwente.nl/~woegingergj/papers/ptas.ps
[11] S. Sevastianov, Personal communication, 2005.